

**PRACTICAL NUMBER: 1****Aim: Design an Expert system using AIML****E.g: An expert system for responding the patient query for identifying the flu.****Code:**

```
import aiml
```

```
kernel=aiml.Kernel()
```

```
kernel.learn("std-startup.xml")
```

```
kernel.respond("load aiml b")
```

```
while True:
```

```
    input_text=input(">Human:")
```

```
    response=kernel.respond(input_text)
```

```
    print(">Bot: "+response)
```

**std-startup.xml**

```
<aiml version="2.0" encoding="UTF-8">
```

```
    <!-- std-startup.xml -->
```

```
    <category>
```

```
        <!-- Pattern to match in user input -->
```

```
        <!-- If user enters "Load AIML B" -->
```

```
        <pattern>LOAD AIML B</pattern>
```

```
        <!-- Template is the response to the pattern -->
```

```
        <!-- This learns an aiml file -->
```

```
        <template>
```

```
            <learn>basic_chat.aiml</learn>
```

```
            <!-- You can add more aiml files here -->
```

```
            <!--<learn>more_aiml.aiml</learn>-->
```

```
        </template>
    </category>
</aiml>
```

**basic\_chat.aiml**

```
<aiml version="1.0.1" encoding="UTF-8">
```

```
<!-- basic_chat.aiml -->
```

```
    <category>
        <pattern>HELLO *</pattern>
        <template>
            Well, hello I am your friendly nurse chatbot!
            How are you feeling today?
        </template>
    </category>
```

```
    <category>
        <pattern>GOOD</pattern>
        <template>
            Ok great! Have a nice day.
        </template>
    </category>
```

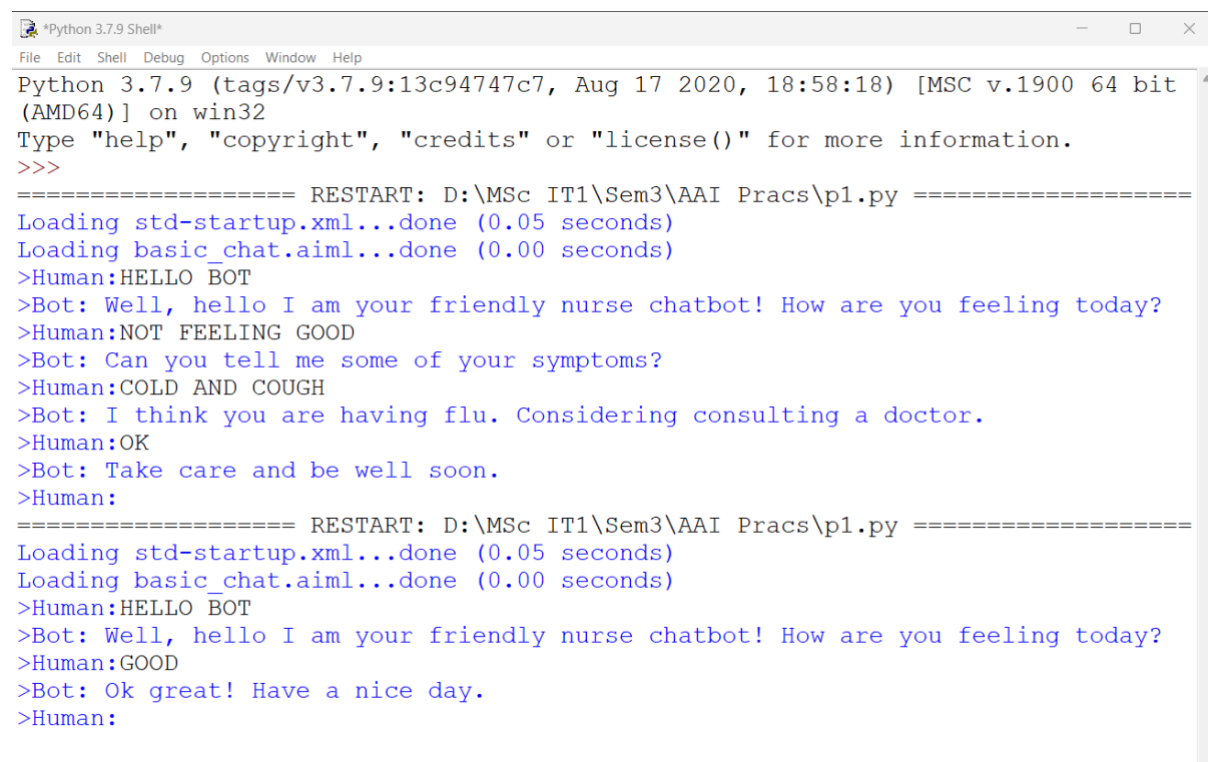
```
    <category>
        <pattern>NOT FEELING GOOD</pattern>
        <template>
            Can you tell me some of your symptoms?
        </template>
    </category>
```

```

<category>
    <pattern>COLD AND COUGH</pattern>
    <template>
        I think you are having flu. Considering consulting a doctor.
    </template>
</category>

<category>
    <pattern>OK</pattern>
    <template>
        Take care and be well soon.
    </template>
</category>
</aiml>

```

**Output:**


```

Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\MSc IT1\Sem3\AAI Pracs\p1.py =====
Loading std-startup.xml...done (0.05 seconds)
Loading basic_chat.aiml...done (0.00 seconds)
>Human:HELLO BOT
>Bot: Well, hello I am your friendly nurse chatbot! How are you feeling today?
>Human:NOT FEELING GOOD
>Bot: Can you tell me some of your symptoms?
>Human:COLD AND COUGH
>Bot: I think you are having flu. Considering consulting a doctor.
>Human:OK
>Bot: Take care and be well soon.
>Human:
===== RESTART: D:\MSc IT1\Sem3\AAI Pracs\p1.py =====
Loading std-startup.xml...done (0.05 seconds)
Loading basic_chat.aiml...done (0.00 seconds)
>Human:HELLO BOT
>Bot: Well, hello I am your friendly nurse chatbot! How are you feeling today?
>Human:GOOD
>Bot: Ok great! Have a nice day.
>Human:

```

**PRACTICAL NUMBER: 2****Aim: Design a bot using AIML.****Code:**

```
import aiml

kernel = aiml.Kernel()

kernel.learn("std2-startup.xml")

kernel.respond ("load prac 2")

while True:

    input_text = input("> Human: ")

    response = kernel.respond(input_text)

    print("> Bot: "+response)
```

**std2-startup.xml**

```
<aiml version="1.0.1" encoding="UTF-8">

  <!--std-startup.xml -->

  <category>

    <!-- Pattern to match in user input -->

    <!-- If user enters "LOAD AIML B" -->

    <pattern>LOAD PRAC 2</pattern>

    <!-- Template is the response to the pattern -->

    <!-- This learn an aiml file -->

    <template>

      <learn>prac2_chat.aiml</learn>

      <!-- You can add more aiml files here --> <!--<learn>more_aiml.aiml</learn>-->

    </template>

  </category>
```

</aiml>

### **prac2\_chat.aiml**

<aiml version="1.0.1" encoding="UTF-8">

<!-- prac2\_chat.aiml -->

<category>

<pattern> HELLO \* </pattern>

<template>

Hello user! Enter any day of the week.

</template>

</category>

<category>

<pattern>SUNDAY</pattern>

<template>comes after Saturday and before Monday.</template>

</category>

<category>

<pattern>MONDAY</pattern>

<template>comes after Sunday and before Tuesday.</template>

</category>

<category>

<pattern>TUESDAY</pattern>

<template>comes after Monday and before Wednesday.</template>

</category>

<category>

<pattern>WEDNESDAY</pattern>

<template>comes after Tuesday and before Thursday.</template>

</category>

<category>

<pattern>THURSDAY</pattern>

```
<template>comes after Wednesday and before Friday.</template>

</category>

<category>

    <pattern>FRIDAY</pattern>

    <template>comes after Thursday and before Saturday.</template>

</category>

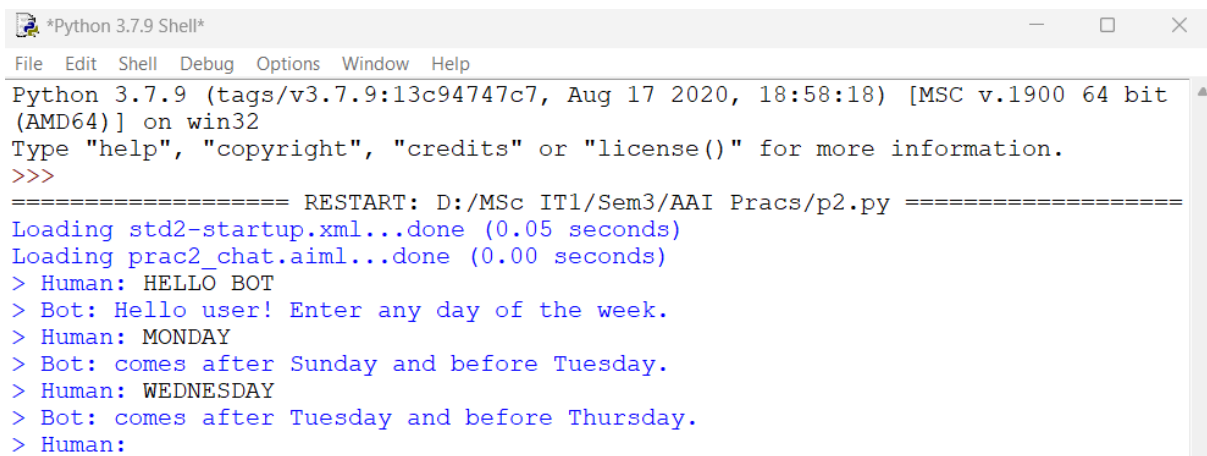
<category>

    <pattern>SATURDAY</pattern>

    <template>comes after Friday and before Sunday.</template>

</category>

</aiml>
```

**Output:**

```
*Python 3.7.9 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/MSc IT1/Sem3/AAI Pracs/p2.py =====
Loading std2-startup.xml...done (0.05 seconds)
Loading prac2_chat.aiml...done (0.00 seconds)
> Human: HELLO BOT
> Bot: Hello user! Enter any day of the week.
> Human: MONDAY
> Bot: comes after Sunday and before Tuesday.
> Human: WEDNESDAY
> Bot: comes after Tuesday and before Thursday.
> Human:
```

**PRACTICAL NUMBER: 3****Aim: Implement Bayes Theorem using Python.****Code:**

```
def diabetes_prob(
    prob_th=0.8,
    sensitivity=0.79,
    specificity=0.79,
    prevelance=0.02,
    verbose=True):

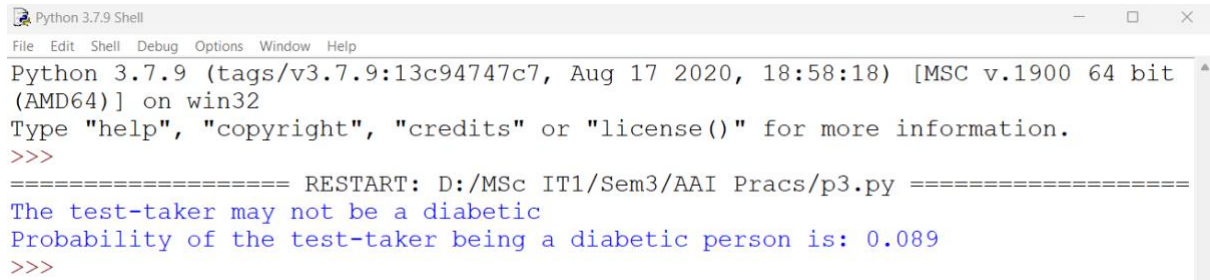
    #Computes the posterior using Bayes rule
    p_user=prevelance
    p_non_user=1-prevelance
    p_pos_user=sensitivity
    p_neg_user=specificity
    p_pos_non_user=1-specificity

    num=p_pos_user*p_user
    den=p_pos_user*p_user+p_pos_non_user*p_non_user

    prob=num/den
    if verbose:
        if prob>prob_th:
            print("The test-taker could be diabetic")
        else:
            print("The test-taker may not be a diabetic")
    return prob

p=diabetes_prob(prob_th=0.5,sensitivity=0.97,specificity=0.95,prevelance=0.005)
```

```
print("Probability of the test-taker being a diabetic person is:",round(p,3))
```

**Output:**A screenshot of a Python 3.7.9 Shell window. The window title is "Python 3.7.9 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output:

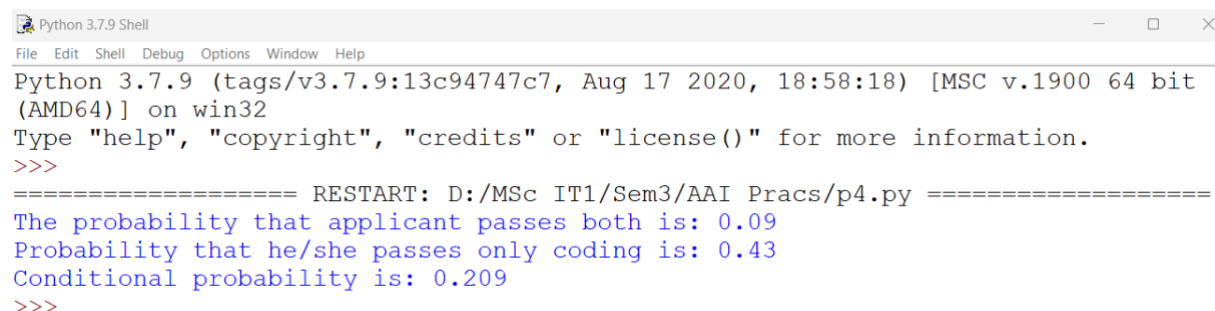
```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: D:/MSc IT1/Sem3/AAI Pracs/p3.py =====  
The test-taker may not be a diabetic  
Probability of the test-taker being a diabetic person is: 0.089  
>>>
```



**PRACTICAL NUMBER: 4****Aim: Implement Conditional Probability and joint probability using Python.****Code:****Conditional probability:**

```
def conditional():  
    pass_stats=0.15  
    pass_codingWStats=0.60  
    pass_codingWOStats=0.40  
    prob_both=pass_stats*pass_codingWStats  
    print("The probability that applicant passes both is:",round(prob_both,3))  
    prob_coding=(prob_both)+((1-pass_stats)*pass_codingWOStats)  
    print("Probability that he/she passes only coding is:",round(prob_coding,3))  
    stats_given_coding=prob_both/prob_coding  
    print("Conditional probability is:",round(stats_given_coding,3))
```

conditional()

**Output:**

```
Python 3.7.9 Shell  
File Edit Shell Debug Options Window Help  
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit  
(AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: D:/MSc IT1/Sem3/AAI Pracs/p4.py =====  
The probability that applicant passes both is: 0.09  
Probability that he/she passes only coding is: 0.43  
Conditional probability is: 0.209  
>>>
```

**Joint probability:**

```
import enum, random  
  
class Kid(enum.Enum):  
    BOY = 0  
    GIRL = 1  
  
def random_kid() -> Kid:
```

```
return random.choice([Kid.BOY, Kid.GIRL])

# We create variables representing joint distributions; one variable for both children being
girls (both_girls), one variable for only the older child being a girl (older_girl), and one for at
least one child being a girl (either_girl).

both_girls = 0

older_girl = 0

either_girl = 0

random.seed(0)

for _ in range(10000):

    younger = random_kid()

    older = random_kid()

    if older == Kid.GIRL:

        older_girl += 1

    if older == Kid.GIRL and younger == Kid.GIRL:

        both_girls += 1

    if older == Kid.GIRL or younger == Kid.GIRL:

        either_girl += 1

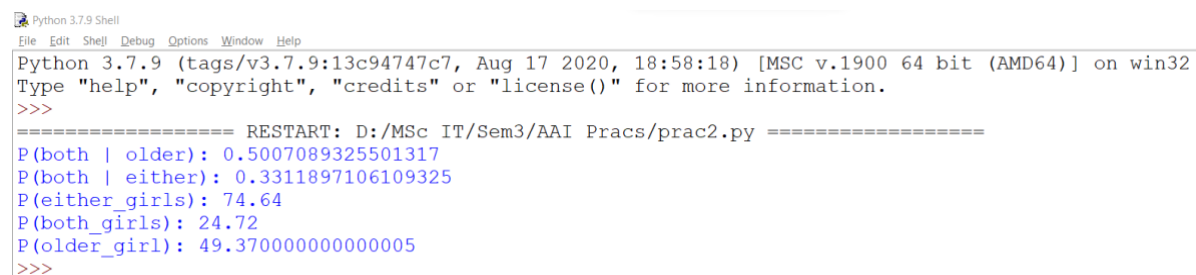
print("P(both | older):", both_girls / older_girl)

print("P(both | either):", both_girls / either_girl)

print("P(either_girls):", (either_girl/10000)*100)

print("P(both_girls):", (both_girls/10000)*100)

print("P(older_girl):", (older_girl/10000)*100)
```

**Output:**

```
Python 3.7.9 Shell
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/MSc IT/Sem3/AAI Pracs/prac2.py =====
P(both | older): 0.5007089325501317
P(both | either): 0.3311897106109325
P(either_girls): 74.64
P(both_girls): 24.72
P(older_girl): 49.370000000000005
>>>
```

**PRACTICAL NUMBER: 5**

**Aim: Write a program for to implement Rule based system.**

**Code:**

```
import contextlib
import sys

from pyke import knowledge_engine
from pyke import krb_traceback

engine = knowledge_engine.engine(__file__)

def test():
    engine.reset()    # Allows us to run tests multiple times.

    engine.activate('simple_rules') #STUDENTS: you will need to edit the name of your rule
    file here

    print("doing proof")

    try:
        with engine.prove_goal('simple_rules.what_to_bring($bring)') as gen: #STUDENTS: you
            will need to edit this line

            for vars, plan in gen:

                print("You should bring: %s" % (vars['bring'])) #STUDENTS: you will need to edit this
                line

    except Exception:

        # This converts stack frames of generated python functions back to the
        # .krb file.

        krb_traceback.print_exc()

        sys.exit(1)

    print()

    print("done")

    #engine.print_stats()
```

```

def test_questions():
    engine.reset()    # Allows us to run tests multiple times.

    engine.activate('simple_rule_questions') #STUDENTS: you will need to edit the name of
your rule file here

    print("doing proof")

    try:

        with engine.prove_goal('simple_rule_questions.what_to_bring($bring)') as gen:
#STUDENTS: you will need to edit this line

            for vars, plan in gen:

                print("You should bring: %s" % (vars['bring'])) #STUDENTS: you will need to edit this
line

    except Exception:

        # This converts stack frames of generated python functions back to the
        # .krb file.

        krb_traceback.print_exc()

        sys.exit(1)

    print()

    print("done")

test()

test_questions()

Files:

facts.kfb:

#Weather facts

raining(True)

windy(True)

simple_rules.krb

# simple_rules.krb

wear_rain_protection

    use rain_protection(True)

```

when

facts.raining(True)

what\_to\_bring\_raincoat

use what\_to\_bring(raincoat)

when

rain\_protection(\$protection)

facts.windy(True)

what\_to\_bring\_umbrella

use what\_to\_bring(umbrella)

when

rain\_protection(\$protection)

facts.windy(False)

what\_to\_bring\_nothing

use what\_to\_bring(nothing)

when

facts.raining(False)

facts.windy(False)

### **questions.kqb**

# questions.kqb

any\_disasters(\$ans)

Are any of the following disasters currently occurring?

---

\$ans = select\_1

1: Forest Fire

2: Tornado

3: Hurricane

4: Pandemic

5: None of the above

! Thank goodness

```
is_raining($ans)
```

```
    Is it raining?
```

```
    ---
```

```
    $ans = yn
```

```
is_windy($ans)
```

```
    Is it windy?
```

```
    ---
```

```
    $ans = yn
```

```
simple_rule_questions.krb
```

```
# simple_rule_questions.krb
```

```
no_rain
```

```
    use what_to_bring(no_rain_gear)
```

```
    when
```

```
        questions.is_raining(False)
```

```
what_to_bring_raincoat
```

```
    use what_to_bring(raincoat)
```

```
    when
```

```
        questions.is_raining(True)
```

```
        questions.is_windy(False)
```

```
what_to_bring_umbrella
```

```
    use what_to_bring(umbrella)
```

```
    when
```

```
questions.is_raining(True)
```

```
questions.is_windy(True)
```

```
what_to_bring_safety_gear
```

```
use what_to_bring(safety_gear)
```

```
when
```

```
questions.any_disasters($ans)
```

```
check $ans in (1,2,3)
```

```
what_to_bring_tissues
```

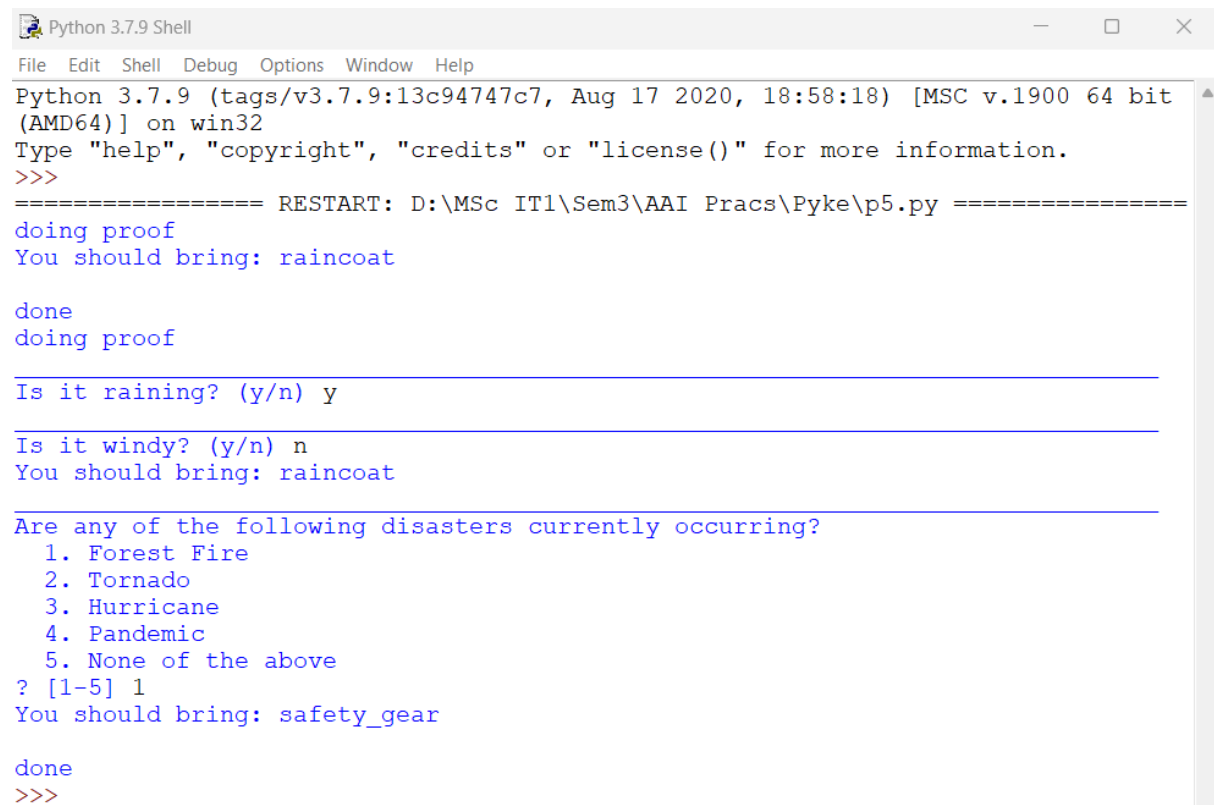
```
use what_to_bring(tissues)
```

```
when
```

```
questions.any_disasters($ans)
```

```
check $ans in (4,)
```

### Output:



```
Python 3.7.9 Shell
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: D:\MSc IT1\Sem3\AAI Pracs\Pyke\p5.py =====
doing proof
You should bring: raincoat

done
doing proof

Is it raining? (y/n) y

Is it windy? (y/n) n
You should bring: raincoat

Are any of the following disasters currently occurring?
1. Forest Fire
2. Tornado
3. Hurricane
4. Pandemic
5. None of the above
? [1-5] 1
You should bring: safety_gear

done
>>>
```

**PRACTICAL NUMBER: 6****Aim: Design a Fuzzy based application using Python / R.****Code:**

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

quality=ctrl.Antecedent(np.arange(0,11,1),'quality')
service=ctrl.Antecedent(np.arange(0,11,1),'service')
tip=ctrl.Consequent(np.arange(0,26,1),'tip')
quality.automf(3)
service.automf(3)

tip['low']=fuzz.trimf(tip.universe,[0,0,13])
tip['medium']=fuzz.trimf(tip.universe,[0,13,25])
tip['high']=fuzz.trimf(tip.universe,[13,25,25])

quality['average'].view()
service.view()
tip.view()

rule1=ctrl.Rule(quality['poor']|service['poor'],tip['low'])
rule2=ctrl.Rule(service['average'],tip['medium'])
rule3=ctrl.Rule(service['good']|quality['good'],tip['high'])
rule1.view()
tipping_ctrl=ctrl.ControlSystem([rule1,rule2,rule3])
tipping=ctrl.ControlSystemSimulation(tipping_ctrl)
tipping.input['quality']=6.5
```

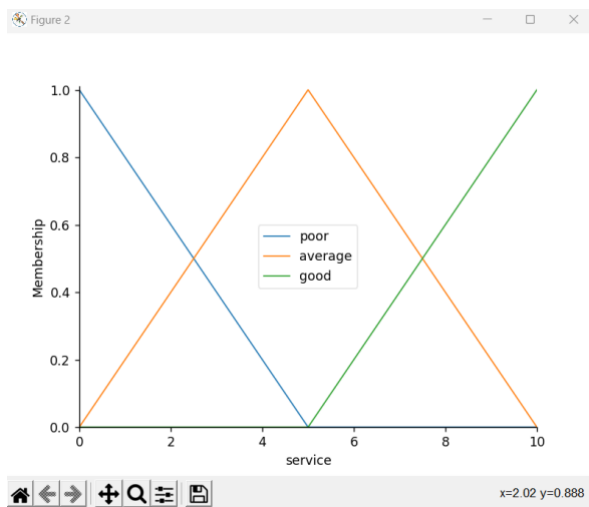
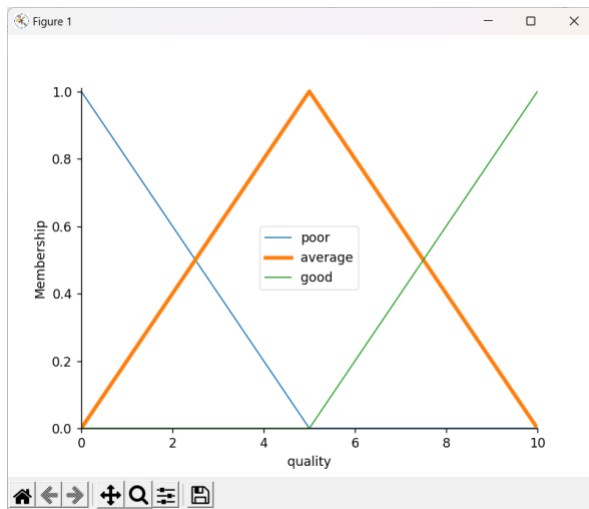


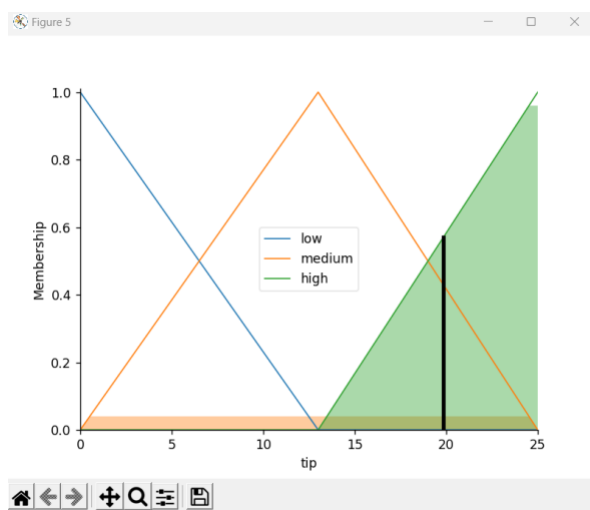
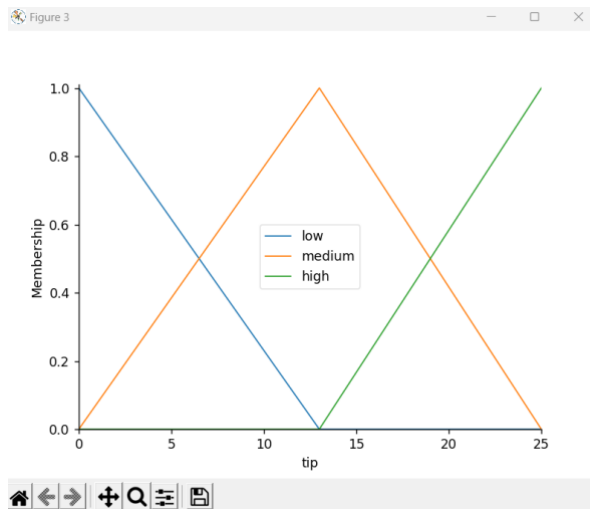
```
tipping.input['service']=9.8
```

```
tipping.compute()
```

```
print(tipping.output['tip'])
```

```
tip.view(sim=tipping)
```

**Output:**



Python 3.7.9 Shell

File Edit Shell Debug Options Window Help

Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: D:\MSc IT1\Sem3\AAI Pracs\p6.py =====

19.847607361963192

>>>

**PRACTICAL NUMBER: 7**

**Aim: Write an application to simulate supervised and un-supervised learning model.**

**Supervised learning:****Code:**

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
import seaborn as sns

#Importing the dataset
dataset=pd.read_csv("iris.csv")
dataset.describe()

X=dataset.iloc[:,[0,1,2,3]].values
y=dataset.iloc[:,4].values
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)

sc=StandardScaler()

X_train=sc.fit_transform(X_train)

X_test=sc.transform(X_test)


#Fitting Logistic Regression to the training set

classifier=LogisticRegression(random_state=0,solver='lbfgs',multi_class='auto')

classifier.fit(X_train,y_train)


#Predicting the test set results

y_pred=classifier.predict(X_test)

#Predict probabilities

probs_y=classifier.predict_proba(X_test)

cm=confusion_matrix(y_test,y_pred)

print('Confusion matrix:\n',cm)

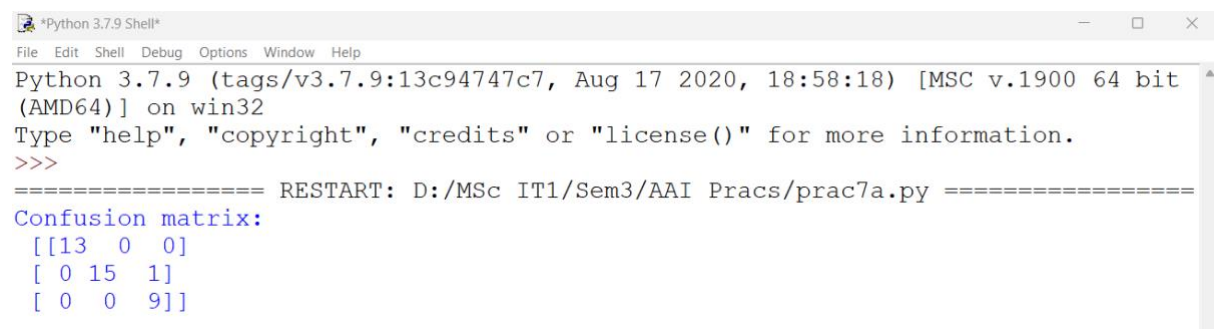

ax=plt.subplot()

df_cm=cm

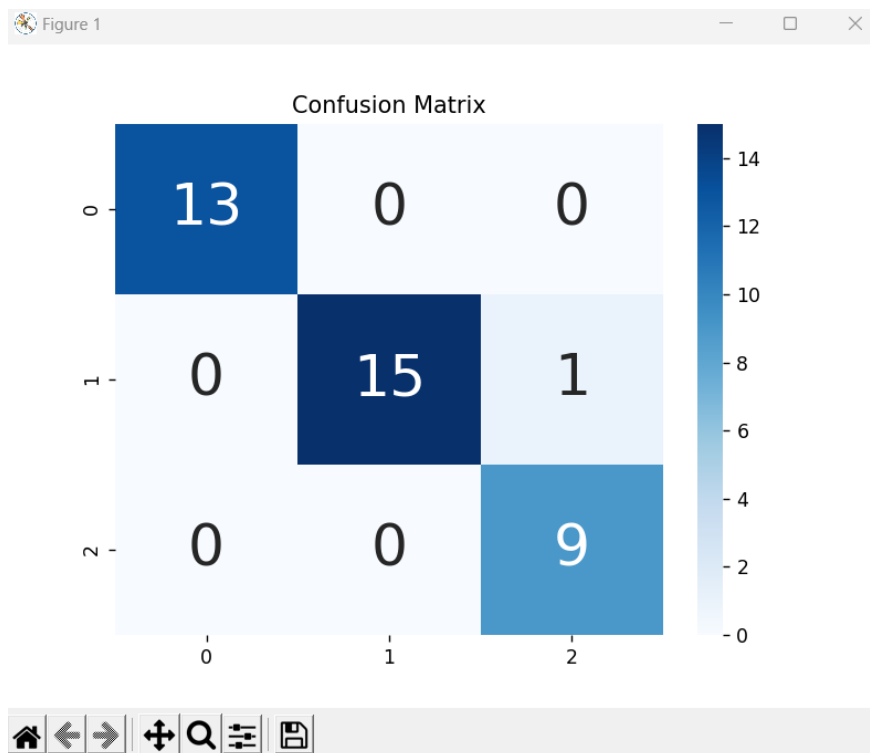
sns.heatmap(df_cm,annot=True,annot_kws={"size":30},fmt="d",cmap="Blues",ax=ax)

ax.set_title('Confusion Matrix')

plt.show()
```

**Output:**

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/MSc IT1/Sem3/AAI Pracs/prac7a.py =====
Confusion matrix:
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```



### Unsupervised learning:

#### Code:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering

customer_data = pd.read_csv('shopping_data.csv')
#shape: Returns tuple of shape (Rows, columns) of dataframe/series
print("Rows,columns in data:",customer_data.shape)
#head: Displays the first five rows of the dataframe by default.
print("First five rows:",customer_data.head())
#iloc: helps us to select a specific row or column from the data set.
data = customer_data.iloc[:,3:5].values
plt.figure(figsize=(10,7))
```

```
plt.title("Customer Dendograms")

# create a dendrogram variable linkage is actually the algorithm itself of hierarchical
clustering and then

# in linkage we have to specify on which data we apply and engage.

# Ward method is actually a method that tries to minimize the variance within each cluster.

# We choose Euclidean distance and ward method for our algorithm class.

dend = shc.dendrogram(shc.linkage(data,method = 'ward'))

cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')

cluster.fit_predict(data)

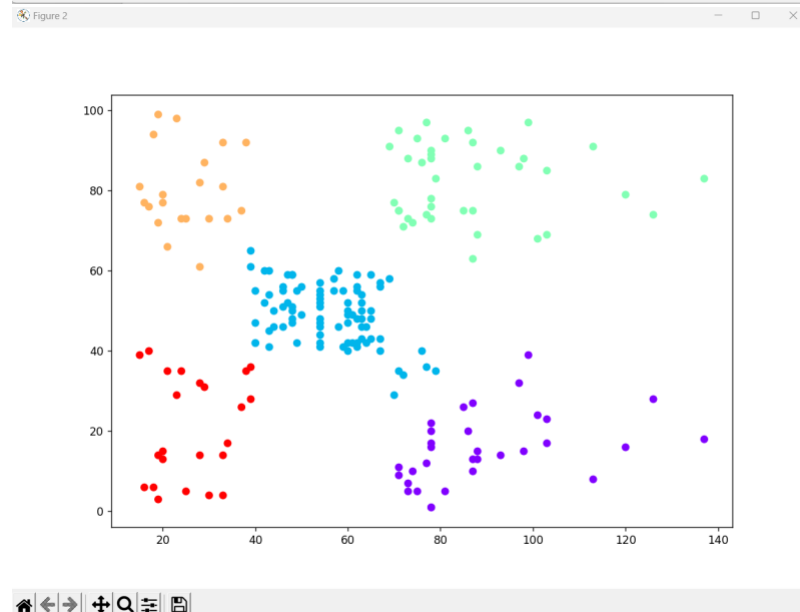
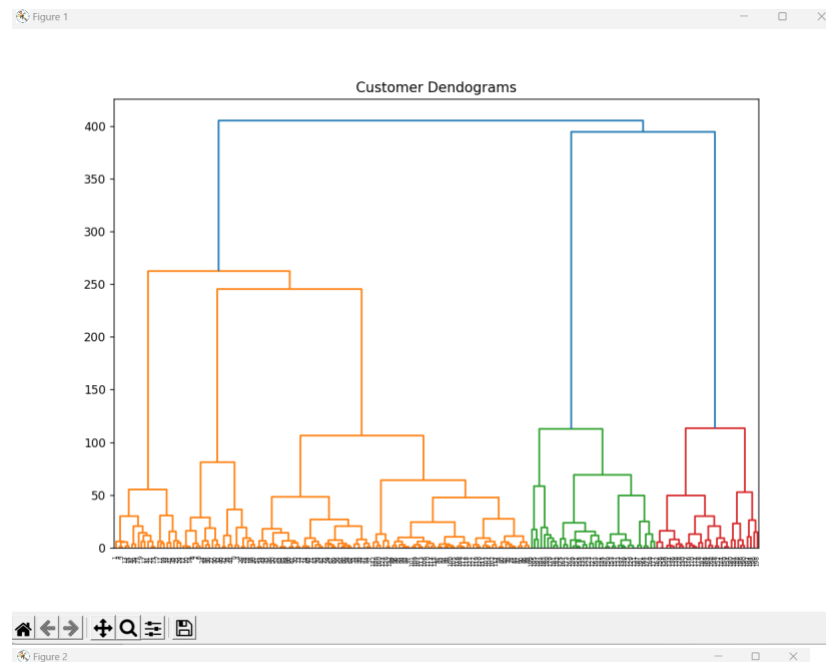
plt.figure(figsize=(10, 7))

plt.scatter(data[:,0], data[:,1], c=cluster.labels_,cmap='rainbow')

plt.show()
```

**Output:**

```
*Python 3.7.9 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/MSc IT1/Sem3/AAI Pracs/p7b.py =====
Rows,columns in data: (200, 5)
First five rows:  CustomerID  Genre  Age  Annual Income (k$)  Spending Score
(1-100)
0          1    Male    19          15          39
1          2    Male    21          15          81
2          3  Female    20          16           6
3          4  Female    23          16          77
4          5  Female    31          17          40
```



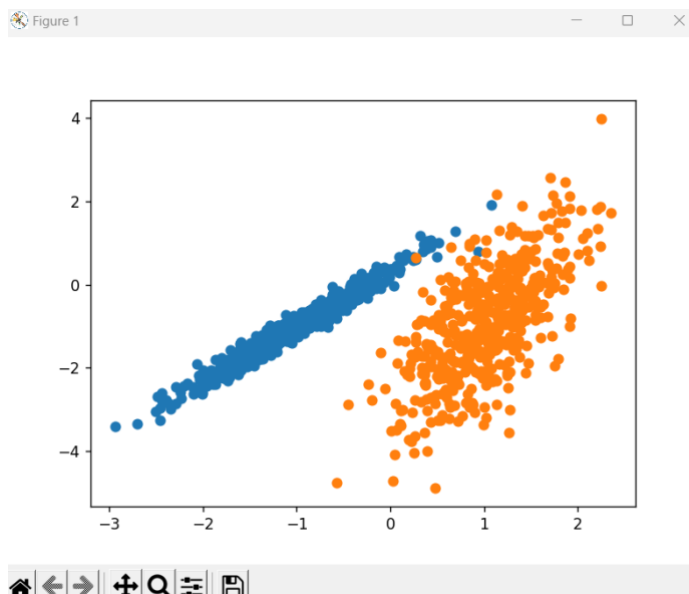
**PRACTICAL NUMBER: 8****Aim: Write an application to implement clustering algorithm.****Code:**

```
from numpy import where
from sklearn.datasets import make_classification
from matplotlib import pyplot

#Define dataset
X,y=make_classification(n_samples=1000,n_features=2,n_informative=2,n_redundant=0,n_
clusters_per_class=1,random_state=4)

#Create scatter plot for samples from each class
for class_value in range(2):
    #Get row indexes for samples with this class
    row_ix=where(y==class_value)
    #Create scatter of these samples
    pyplot.scatter(X[row_ix,0],X[row_ix,1])

#Show the plot
pyplot.show()
```

**Output:**



**PRACTICAL NUMBER: 9****Aim: Write an application to implement support vector machine algorithm.****Code:**

```
# importing scikit learn with make_blobs
from sklearn.datasets.samples_generator import make_blobs
import numpy as np
import matplotlib.pyplot as plt

# creating datasets X containing n_samples
# Y containing two classes
X, Y = make_blobs(n_samples=500, centers=2, random_state=0, cluster_std=0.40)

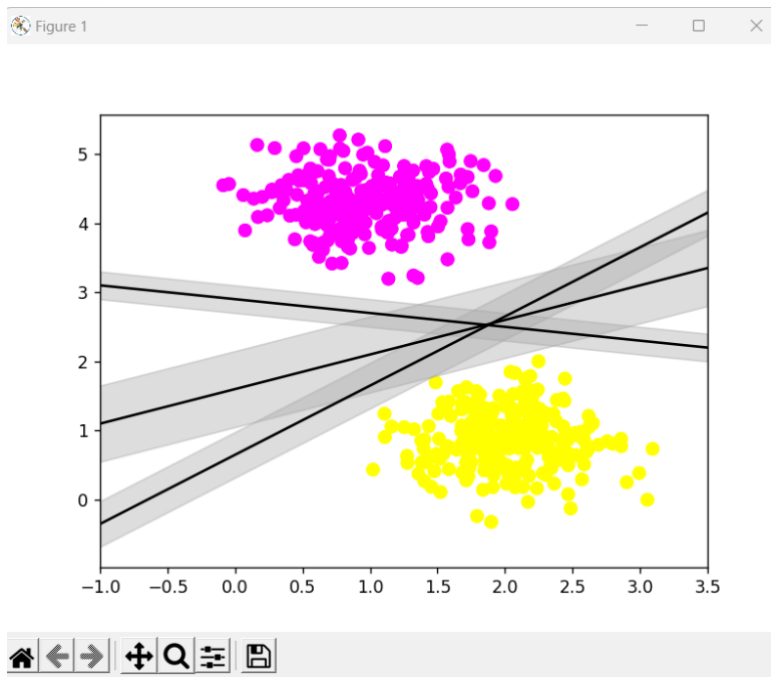
# plotting scatters
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring');

# creating linspace between -1 to 3.5
xfit = np.linspace(-1, 3.5)

# plotting scatter
plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring')

# plot a line between the different sets of data
for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
                     color='#AAAAAA', alpha=0.4)

plt.xlim(-1, 3.5);
plt.show()
```

**Output:**

**PRACTICAL NUMBER:10**

**Aim: Simulate genetic algorithm with suitable example using Python / R or any other platform.**

**Code:**

```
from __future__ import print_function
import numpy as num
from sklearn import datasets, linear_model
from genetic_selection import GeneticSelectionCV

def main():
    iris = datasets.load_iris()

    # Some noisy data not correlated
    e = num.random.uniform(0, 0.2, size=(len(iris.data), 30))
    x = num.hstack((iris.data, e))
    Y = iris.target

    estimators = linear_model.LogisticRegression(solver="liblinear", multi_class="ovr")
    selectors = GeneticSelectionCV(estimators,
                                  cv=6,
                                  verbose=2,
                                  scoring="accuracy",
                                  max_features=6,
                                  n_population=60,
                                  crossover_proba=0.6,
                                  mutation_proba=0.2,
                                  n_generations=50,
                                  crossover_independent_proba=0.6,
                                  mutation_independent_proba=0.06,
                                  tournament_size=4,
```

```

n_gen_no_change=20,

caching=True,

n_jobs=-2)

selectors = selectors.fit(x, Y)

print(selectors.support_)

if __name__ == "__main__":

    main()

```

### Output:

```

Python 3.7.9 Shell
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\MSc IT1\Sem3\AAI Pracs\pl0.py =====
Selecting features with genetic algorithm.
gen nevals avg std min max
0 60 [ 0.462556 3.083333 0.067643] [ 0.210843 1.705791 0.021153] [ 0.186667 1. 0.029814] [ 0.92 6. 0.127366]
1 36 [-332.634556 3.933333 333.398174] [ 1795.184704 1.671991 1795.042895] [-10000. 0. 0.04] [ 0.92 6.
10000. ]
2 46 [-1832.663111 4.833333 1833.379004] [ 3869.713145 2.114763 3869.37395 ] [-10000. 1. 0.04] [ 0.92
11. 10000. ]
3 41 [-665.828778 4.616667 666.707792] [ 2494.662194 1.212321 2494.427267] [-10000. 2. 0.04] [ 0.95333
3 8. 10000. ]
4 38 [-1165.853 4.783333 1166.703861] [ 3210.522419 1.592604 3210.213197] [-10000. 3. 0.035901]
[ 0.953333 12. 10000. ]
5 39 [-499.110778 4.6 500.039675] [ 2179.653473 1.083205 2179.44037 ] [-10000. 3. 0.035901]
[ 0.953333 7. 10000. ]
6 45 [-2332.606111 5.466667 2333.365585] [ 4229.927039 1.359739 4229.508054] [-10000. 3. 0.035901]
[ 0.953333 9. 10000. ]
7 45 [-499.097444 4.716667 500.037219] [ 2179.656532 0.932589 2179.440933] [-10000. 3. 0.035901]
[ 0.953333 8. 10000. ]
8 44 [-332.413778 4.133333 333.369554] [ 1795.225693 0.805536 1795.04821 ] [-10000. 3. 0.035901]
[ 0.953333 7. 10000. ]
9 38 [-332.413778 3.816667 333.369435] [ 1795.225693 0.974537 1795.048232] [-10000. 2. 0.035901]
[ 0.96 7. 10000. ]
10 30 [-832.460667 3.966667 833.367571] [ 2764.117111 1.538036 2763.843669] [-10000. 3. 0.035901]
[ 0.96 9. 10000. ]
11 35 [-832.459 4.55 833.368905] [ 2764.117613 1.697302 2763.843267] [-10000. 3. 0.035901]
[ 0.96 11. 10000. ]
12 42 [-1499.190889 4.983333 1500.034199] [ 3571.054109 1.384337 3570.699848] [-10000. 3. 0.035901]
[ 0.96 9. 10000. ]
13 44 [-665.774111 5.2 666.704532] [ 2494.676803 0.702377 2494.428138] [-10000. 3. 0.035901]
[ 0.96 7. 10000. ]
14 33 [-665.772222 5.25 666.704237] [ 2494.677308 0.849019 2494.428217] [-10000. 4. 0.035901]
[ 0.96 9. 10000. ]
15 39 [-1665.868111 5.55 1666.700302] [ 3727.137087 1.309262 3726.76492 ] [-10000. 3. 0.04]
[ 0.96 11. 10000. ]
16 36 [-332.407 5.05 333.372166] [ 1795.226952 0.529937 1795.047725] [-10000. 3. 0.035901]
[ 0.96 7. 10000. ]
17 37 [-665.773111 5.2 666.704614] [ 2494.677071 0.6 2494.428116] [-10000. 5. 0.04]
[ 0.96 8. 10000. ]
18 36 [-499.088222 5.133333 500.03807 ] [ 2179.658648 0.531246 2179.440738] [-10000. 5. 0.04]
[ 0.96 8. 10000. ]
19 37 [-665.775444 5.3 666.705128] [ 2494.676447 0.842615 2494.427979] [-10000. 5. 0.04]
[ 0.96 9. 10000. ]
20 43 [-832.454222 5.233333 833.370257] [ 2764.119054 0.642045 2763.842859] [-10000. 5. 0.04]
[ 0.96 8. 10000. ]
21 40 [-832.454556 5.35 833.37033 ] [ 2764.118953 1.029968 2763.842837] [-10000. 5. 0.035901]
[ 0.96 11. 10000. ]
22 40 [-665.770889 5.216667 666.704074] [ 2494.677665 0.709264 2494.42826 ] [-10000. 5. 0.035901]
[ 0.96 8. 10000. ]
23 39 [-165.724222 5.083333 166.706524] [ 1280.313654 0.525727 1280.185769] [-10000. 5. 0.04]
[ 0.96 9. 10000. ]
24 40 [-999.136333 5.25 1000.036006] [ 3000.287889 0.744424 2999.987998] [-10000. 4. 0.035901]
[ 0.96 8. 10000. ]
25 39 [-1165.824556 5.4 1166.703725] [ 3210.532756 0.916515 3210.213246] [-10000. 5. 0.04]
[ 0.96 10. 10000. ]
26 46 [-665.771444 5.316667 666.70415 ] [ 2494.677516 1.008161 2494.42824 ] [-10000. 5. 0.035901]
[ 0.96 12. 10000. ]
27 43 [-499.092222 5.216667 500.038587] [ 2179.65773 0.877338 2179.440619] [-10000. 3. 0.035901]
[ 0.96 9. 10000. ]
28 39 [-665.772111 5.25 666.704611] [ 2494.677338 0.744424 2494.428117] [-10000. 5. 0.04]
[ 0.96 9. 10000. ]
29 49 [-832.453778 5.316667 833.369937] [ 2764.119188 1.1029 2763.842955] [-10000. 4. 0.035901]
[ 0.96 12. 10000. ]
[ True False True True False False False False True True False
False False False False False False False False False False
False False False False False False False False False]
>>>

```